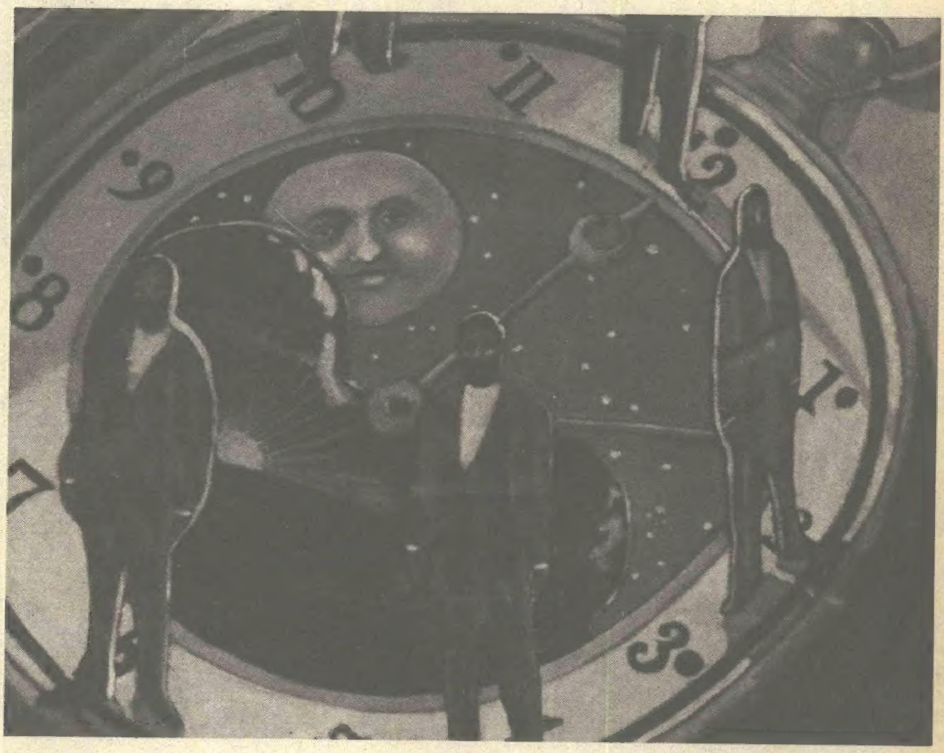
จเร เลิศสุดวิชัย

พัฒนาโปรแกรมยูทิลิตี้ด้วย เทอร์โบซี

: ตอนที่ 2

**เสนอโปรแกรม
ยูทิลิตี้ที่น่าสนใจ
เพื่อให้ท่านผู้อ่าน
ดัดแปลงหรือเพิ่มความ
สามารถได้ตามใจชอบ**



ในฉบับที่แล้วมุมภาษาซีได้เสนอโปรแกรมยูทิลิตี้เพื่อช่วยงานดอสไปแล้วสามโปรแกรม ซึ่งท่านผู้อ่านจะเห็นได้ว่าโปรแกรมเหล่านั้นไม่ได้มีการทำงานที่ซ้ำซ้อนจนเกินไป แต่ก็สามารถใช้ช่วยทำงานได้ในยามจำเป็น ซึ่งจากโปรแกรมดังกล่าวผู้อ่านสามารถนำไปดัดแปลงหรือเพิ่มเติมความสามารถด้านอื่น ๆ ได้อีกตามชอบใจ และสำหรับในฉบับนี้ก็เช่นกัน มุมภาษาซีจะเสนอโปรแกรมยูทิลิตี้ที่น่าสนใจอีกเช่นเคยครับ

โปรแกรม LD (LIST DIRECTORY)

การทำงานของโปรแกรม

โปรแกรมจะอ่านเอารายชื่อไดเรคทอรีทั้งหมดที่มีอยู่ในฮาร์ดดิสค์ที่ต่ออยู่กับเครื่องมาแสดงบนจอภาพ (ตั้งแต่ฮาร์ดดิสค์ไดรฟ์ C เป็นต้นไป) ซึ่งผู้ใช้โปรแกรมสามารถที่จะเปลี่ยนทิศทาง (Redirection) ของผลลัพธ์ และเก็บลงไฟล์หรือพิมพ์ออกทางเครื่องพิมพ์

ก็ได้ รายชื่อต่าง ๆ ของไดเรคทอรีเป็นสิ่งจำเป็นมากทีเดียวในการที่จะจัดการโครงสร้างการเก็บข้อมูลบนฮาร์ดดิสค์ให้เรียบร้อย โดยเฉพาะเวลาที่ต้องสำรองไฟล์ข้อมูล (Backup) จากฮาร์ดดิสค์นั้น รายชื่อไดเรคทอรีจะมีประโยชน์มาก เพราะจะถูกใช้ในการพิจารณาและตัดสินใจการเลือกสำรองข้อมูล

การใช้งานโปรแกรม

รูปแบบการใช้งาน C: \ >ld

ตัวอย่างที่ 1

```
C: \ >ld
```

โครงสร้างไดเรคทอรีหรือรายชื่อของไดเรคทอรีของฮาร์ดดิสค์ทุกไดรฟ์ที่ต่ออยู่ (ตั้งแต่ไดรฟ์ C:) จะถูกนำออกมาแสดงบนจอภาพ

ตัวอย่างที่ 2

```
A: \ >ld > prn
```

ในกรณีเช่นนี้ โครงสร้างไดเรคทอรีหรือรายชื่อไดเรคทอรีจะถูกนำออกพิมพ์ทางเครื่องพิมพ์แทน

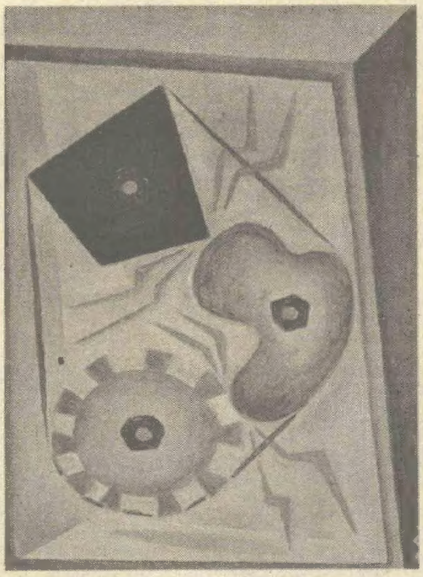
ตัวอย่างที่ 3

```
A: \ >ld > listdir.txt
```

ในกรณีตัวอย่างนี้ รายชื่อไดเรคทอรีจะถูกนำไปเก็บในไฟล์ listdir.txt แทน ซึ่งอาจจะเก็บเอาไว้เพื่อใช้งานในภายหลัง

เทคนิคโปรแกรม

หากผู้อ่านเปรียบเทียบโปรแกรมยูทิลิตี้นี้กับโปรแกรม FS ในตอนที่แล้ว จะพบว่ามีการทำงานที่คล้ายกัน ซึ่งฟังก์ชันไลบรารีที่ใช้งานมากในโปรแกรมก็คือ ฟังก์ชัน findfirst( ) และ findnext( ) โดยจะสังเกตได้ว่าแอตทริบิวต์ของไฟล์ที่ใช้ค้นหาในโปรแกรมนี้จะใช้เพียงแอตทริบิวต์ FA_DIREC คือจะค้นหาแต่ไดเรคทอรีเท่านั้น



โปรแกรม MV (MOVE FILES)

การทำงานของโปรแกรม

โปรแกรม MV จะช่วยในการย้ายกลุ่มไฟล์จากไดเรคทอรีหนึ่งไปยังไดเรคทอรีอื่นตามที่ระบุ โดยจะทำงานได้รวดเร็วและมีประสิทธิภาพ การย้ายกลุ่มไฟล์ของโปรแกรมนี้จะใช้เทคนิคของการเปลี่ยนชื่อไฟล์ด้วยฟังก์ชันในไลบรารีภาษาซี ซึ่งจะช่วยลดขั้นตอนการทำงาน ซึ่งตามปกติแล้วจะต้องทำงานถึง 2 ขั้นตอน โดยขั้นแรกต้องคัดลอก (copy) กลุ่มไฟล์จากไดเรคทอรีต้นทางไปยังไดเรคทอรีปลายทาง แล้วจึงทำการลบกลุ่มไฟล์ที่ไดเรคทอรีต้นทางทิ้งไป ซึ่งจะเห็นว่าทั้งสองขั้นตอนจะต้องใช้เวลาในการทำงานมากทีเดียว การใช้งานโปรแกรม MV สามารถระบุกลุ่มชื่อไฟล์ด้วย wildcards ได้

การใช้งานโปรแกรม

รูปแบบการใช้งาน C: \ >mv filename.ext topath

โดย filename.ext สามารถเป็นได้ทั้งชื่อไฟล์หรือกลุ่มไฟล์โดยใช้ wildcards (คือตัวอักษร '?' และ '*') การระบุชื่อไฟล์ต้นทางสามารถระบุไดเรคทอรีได้ โดยต้องระบุอย่างชัดเจน (ใช้ wildcards ไม่ได้) ตัวอย่างเช่น

```
C: \ >mv C: \ dbase \ dbf \ *.dbf
C: \ dbase \ data
```

สำหรับ topath คือ ไดเรคทอรีปลายทางที่จะย้ายกลุ่มไฟล์ไปเก็บ โดยจะต้องระบุให้ชัดเจน บอกทั้งชื่อไดเรคทอรีให้ครบถ้วน ดังตัวอย่างที่ถูกต้อง

```
C: \ mv*.c   C: \ tc \
C: \ >mv letter.doc   C: \ cw
```

การระบุไดเรคทอรีปลายทางห้ามใส่ชื่อไฟล์ลงไปด้วย พิจารณาดูตัวอย่างที่ผิด ซึ่งโปรแกรมจะทำงานไม่ได้ ดังนี้

```
C: \ >mv*.c   C: \ tc \ *.*
```

เนื่องจากโปรแกรมนี้เป็นการเรียกฟังก์ชัน rename( ) ของเทอร์โบซี ซึ่งการทำงานจะเป็นเพียงการเปลี่ยนชื่อไฟล์เท่านั้น ดังนั้นข้อจำกัดของโปรแกรมนี้คือ จะสามารถย้ายไฟล์เฉพาะบนไดรฟ์เดียวกันเท่านั้น แต่ไม่สามารถย้ายไฟล์ข้ามไดรฟ์ได้ ดังตัวอย่างที่ผิด ซึ่งโปรแกรมจะไม่ทำงาน ดังนี้

```
C: \ >m *.c d: \ tc \
```

เพราะเนื่องจากการทำงานของโปรแกรมดังกล่าวไม่ได้มีการย้ายไฟล์จริง ๆ เกิดขึ้น เพียงแต่ทำการเปลี่ยนชื่อไฟล์เท่านั้น โปรแกรมจึงไม่สามารถทำงานได้

จุดเด่นของโปรแกรมนี้นอกจากเรื่องความเร็วในการทำงานแล้ว โปรแกรมยังช่วยลดการเก็บไฟล์ซ้ำซ้อนได้

ด้วย โดยเฉพาะที่ไดเรคทอรีต้นทางหลังจากย้ายแล้วจะไม่ปรากฏกลุ่มไฟล์ที่ถูกย้ายในไดเรคทอรีอีก ถึงแม้พื้นที่ฮาร์ดดิสค์หรือฟลอปปี้ดิสค์เหลือจำนวนน้อย ซึ่งโปรแกรมนี้ก็ยังสามารถทำงานได้ (ผิดกับวิธีการที่ใช้การคัดลอกแล้วลบออก โดยจะต้องมีพื้นที่ดิสค์ให้เพียงพอกับขนาดไฟล์ทั้งหมดที่ย้าย ซึ่งถ้าหากหาพื้นที่บนดิสค์ไม่พบ ก็จะไม่สามารถทำได้)

ตัวอย่างที่ 1

```
C: \ cw \ >mv letter.doc C: \ cw \ doc
```

เป็นการย้ายไฟล์ letter.doc ไปยังไดเรคทอรี \ cw \ doc

ตัวอย่างที่ 2

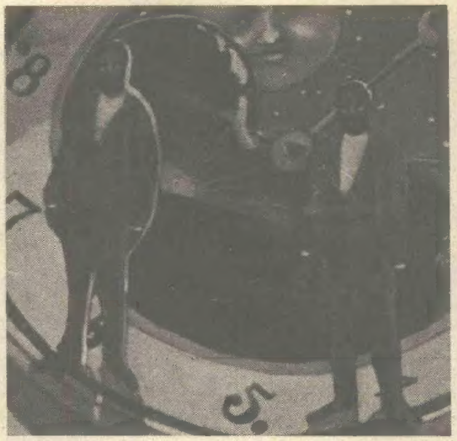
```
C: \ tc \ >mv *.c C: \ tc \ source
```

เป็นการย้ายไฟล์โปรแกรมภาษาซีทุกไฟล์ที่นามสกุล .c ไปยังไดเรคทอรี \ tc \ source

ตัวอย่างที่ 3

```
A: \ mv *.c C: \ tc \
```

ตัวอย่างนี้ต้องการจะย้ายไฟล์ภาษาซีจากไดรฟ์ A: ไปยังไดรฟ์ C: ไดเรคทอรี \ tc \ โปรแกรมจะทำงานไม่ได้ โดยจะพิมพ์ข้อความบอกว่าไม่สามารถย้ายไฟล์เหล่านั้นไปได้ ซึ่งเป็นข้อจำกัดของโปรแกรมดังที่เคยกล่าวมาแล้ว



ตัวอย่างที่ 4

```
C: \ >mv C: \ tc \ source \ *.c
C: \ msc \ source \
```

เป็นการย้ายไฟล์ภาษาซีนามสกุล .c จากไดเรคทอรี \ tc \ source \ ไปยังไดเรคทอรี \ msc \ source

ตัวอย่างที่ 5

```
C: \ >mv
```

ถ้าเรียกชื่อโปรแกรมอย่างเดียว โปรแกรมจะพิมพ์วิธีการใช้งานโปรแกรมให้

เทคนิคโปรแกรม

การทำงานของโปรแกรมนี้ส่วนสำคัญจะอยู่ในฟังก์ชัน move( ) ส่วนในฟังก์ชัน main( ) นั้น เมื่อเริ่มต้นจะนำพารามิเตอร์ตัวแรกซึ่งเป็นชื่อไฟล์ที่ถูกระบุและต้องการย้ายมาแยกออกเป็นข้อมูลรายละเอียดของไฟล์ด้วยฟังก์ชัน fnsplit( ) ซึ่งข้อมูลในส่วนของชื่อไดรฟ์และชื่อไดเรคทอรีจะถูกนำไปใช้เพื่อสร้างเป็นชื่อไฟล์จริง ๆ ที่ต้องการย้ายในฟังก์ชัน move( ) จากนั้นก็จะนำเอาชื่อพาธหรือไดเรคทอรีปลายทางมาเก็บไว้เพื่อนำไปใช้ในการสร้างเป็นชื่อไฟล์ปลายทางที่จะย้ายในฟังก์ชัน move( )

ในฟังก์ชัน move( ) ก็จะเริ่มต้นด้วยการค้นหากลุ่มชื่อไฟล์ที่ระบุด้วยฟังก์ชัน findfirst( ) ซึ่งถ้าหากหาพบก็จะเข้าวนรอบเพื่อเปลี่ยนชื่อไฟล์แต่ละไฟล์โดยสร้างชื่อไฟล์ต้นทางและชื่อไฟล์ปลายทางขึ้น นำชื่อทั้งสองส่งผ่านให้ฟังก์ชัน rename( ) หากการทำงานของฟังก์ชัน rename( ) มีปัญหา ก็จะพิมพ์ข้อความระบุว่าไม่สามารถจะย้ายไฟล์ได้ จากนั้นจะค้นหาชื่อกลุ่มไฟล์ต่าง ๆ ต่อ ด้วยฟังก์ชัน findnext( ) ดังรายละเอียดในโปรแกรมต้นฉบับที่ให้ไว้ และก็เช่นเคยครับสำหรับท่านที่ต้องการโปรแกรมก็หาจากบริการซอฟต์เมล์ฉบับนี้ครับ

รูปที่ 1 โปรแกรม LDIR C

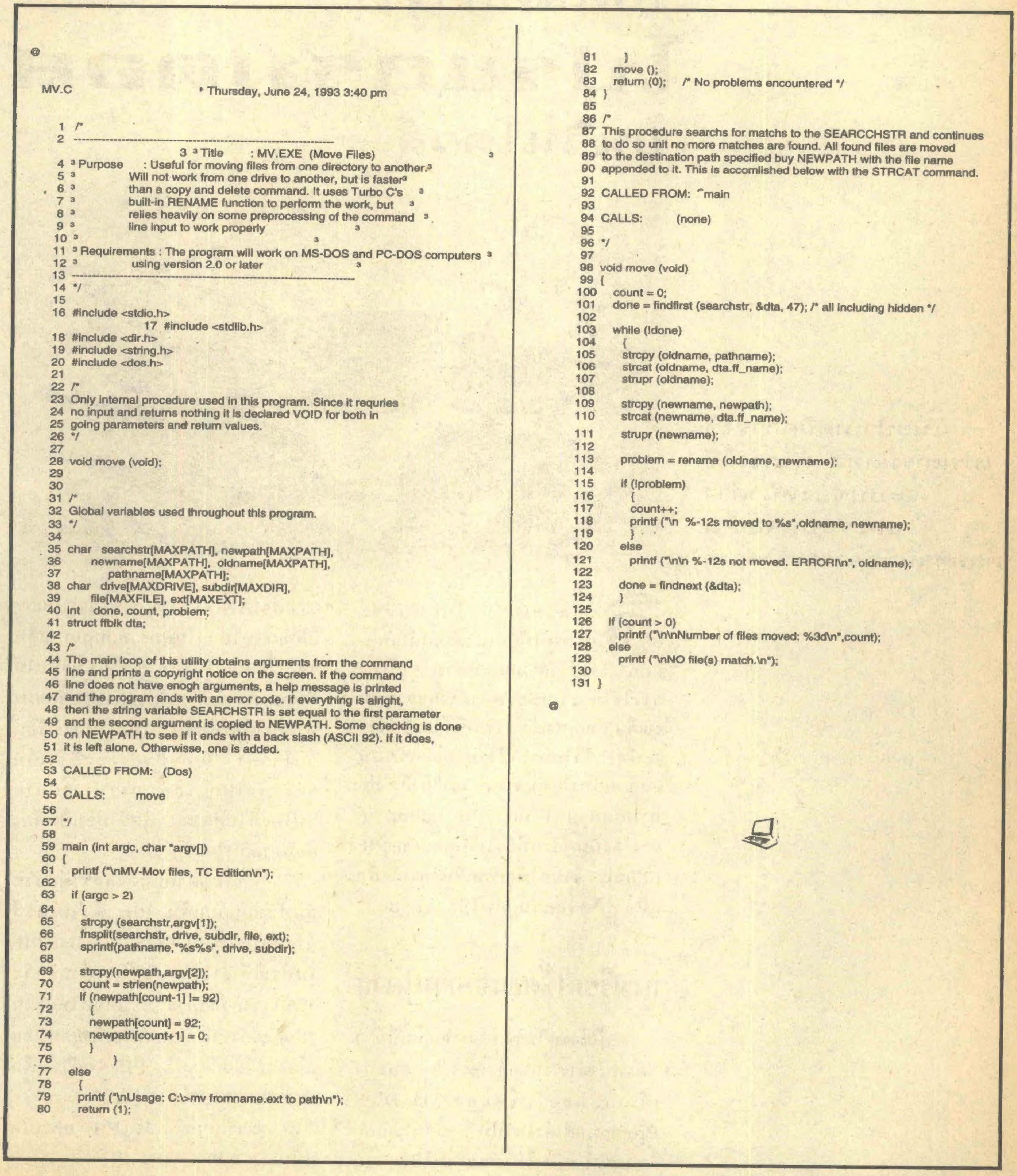
```
@

LDIR.C                         Thursday, June 24, 1993 3:38 pm

/*
-----------------------------------------------------------------
³ Title        : LD.EXE  (List Directory)                       ³
³ Purpose        : With this utility you can kist all the directories   ³
³              which you have access to. It searchs all drives        ³
³              and gives you a complete listing of each directory     ³
³              and subdirectory. The data and time is also given      ³
³              in the output. You can redirect the output to the      ³
³              the printer for documentation of your system.          ³
³                                                    ³
³ Requirements : The program will work on MS-DOS and PC-DOS computers ³
³               using version 2.0 or later                    ³
--------------------------------------------------------------------
*/

#include <stdio.h>
               #include <stdlib.h>
#include <dir.h>
#include <string.h>
#include <dos.h>

/*
The two internal procedures use in this program. The GETDIRECTORY
function returns a pointer to a text string.
*/

char *getdirectory (char *path);
void finddirs (void);

/*
COUNT is used for keeping track of how many directories are
found and MONTHS to print out the month that a directory was        →
```

รูปที่ 1 (ต่อ)

```
created.
*/

int count;
char *months [] =
              {"Jan", "Feb", "Mar", "Apr", "Jun",
               "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

/*
The main loop of this utility does sereral interesting things. One
of these is the creation of a table of valid drives. It assumes that
a system can have 26 drives (one per letter of the alphapet) and
procedes to check the amount of space on each drive. If it gets a
valid number back from the TURBO C function, it assumes that a
drive exists. The scanning for drives is started with the c drive.

CALLED FROM:  (DOS)
CALLS:        finddirs      getdirectory
*/

main ()
{
  /*
  Variables used in the main program.
  */

  char origdir[MAXPATH], rootdir[MAXPATH],
     tempdir[MAXPATH], availdrvs[26];
  unsigned char i;

  struct dfree frespace;

  printf ("\nLD-List Directores,TC Edition\n");

  for (i = 3; i <= 26; i++)       /* Test for the existence of drives */
    {
    getdfree (i, &frespace);
    if (frespace.df_sclus == 0xFFFF) /* Character 205 */
      availdrvs[i] = 0;         /* No drive */
    else
      availdrvs[i] = 1;
    }

  getdirectory (origdir);         /* Get current directory */

  for (i = 3; i <= 26; i++)
    {
    if (availdrvs[i] == 1)
      {
      rootdir[0] = '@' + i;
      rootdir[1] = ':';
      rootdir[2] = '\\';
      rootdir[3] = 0;
      setdisk (i-1);
      getdirectory (tempdir);
      chdir (rootdir);       /* Switch to root */
      count++;
      printf ("\n%3s (root)", rootdir);
      finddirs ();
      chdir (tempdir);
      }
    }
  setdisk (origdir [0] - 'A');
  chdir (origdir);            /* Back to orginal */

  printf ("\n\n%d directories.\n", count);
  return (0);
}



/*
This procedure searchs for directories and once it finds one
prints the date and time that the directory was created on the
screen. It is a recursive routine.

CALLED FROM:   main
CALLS:         getdirectory
*/

void finddirs (void)
{
  /*
  Variables used in the procedure.
  */

  char curdir [MAXPATH],  temp [3],
      AorP[3], fsize[11];

  int  done, junk, attrib,
     month, day, year, hour, minute;

  struct ffblk dta;

  getdirectory (curdir);
  attrib = FA_DIREC + FA_HIDDEN + FA_SYSTEM;
  done = findfirst ("*.*", &dta, attrib);

  while (!done)
    {
    getdirectory (curdir);
    strncpy (temp,dta.ff_name, 2);
    if (((dta.ff_attrib & FA_DIREC) == FA_DIREC) && (temp[0] != '.'))
      {
      /* calculate date and time */
      junk   = dta.ff_fdate;
      day    = junk % 32;
      junk   = junk / 32;
      month  = junk % 16;
      year   = junk / 16 + 1980;
      junk   = dta.ff_ftime / 32;
      minute = junk % 64;
      hour   = junk / 64;

      strcpy (AorP, "am");    /* AM or PM */
      if (hour==12)
        strcpy (AorP, "pm");

      if (hour==0)
        hour = 12;

      if (hour > 12)
        {
        hour = hour - 12;
        strcpy (AorP, "pm");
        }

      count++;

      if (strlen (curdir) != 3)
        strcat (curdir,"\\");

      strcat (curdir,dta.ff_name);
      chdir (curdir);

      printf ("\n%-40s  %2d:%02d%s",curdir,hour,minute,AorP);
      if (month != 0)
         printf ("  %3s %2d, %4d",months[month - 1], day, year);

      finddirs ();
      chdir ("..");
    }
  done = findnext (&dta);
  }
}

/*
This procedure creates a string which holds the druve and path where
the program is searching for backup files. The Turbo C function GETDISK
is called to get the disk drive number. This number is added to the
letter A (the first diskdrive) tro get the letter of the current disk
drive. When the procedure is over, a pointer to drive and path is
return to the caller

CALLED FROM:     main         findfile
CALLS:           (none)
*/

char *getdirectory (char *path)
{
  strcpy  (path, "X:\\");
  path[0] = 'A' + getdisk ();
  getcurdir (0, path + 3);        /* Place it in path area of
                                   (3rd position on) the string */
  return (path);
}


```

■

รูปที่ 2 โปรแกรม MV.C



```
@

MV.C                                  • Thursday, June 24, 1993 3:40 pm

/*
-------------------------------------------------------------
                         ³ Title        : MV.EXE  (Move Files)                  ³
³ Purpose      : Useful for moving files from one directory to another.³
³                  Will not work from one drive to another, but is faster³
³                  than a copy and delete command. It uses Turbo C's    ³
³                  built-in RENAME function to perform the work, but    ³
³                  relies heavily on some preprocessing of the command  ³
³                  line input to work properly                   ³
³                                                  ³
³ Requirements : The program will work on MS-DOS and PC-DOS computers ³
³                  using version 2.0 or later                    ³
---------------------------------------------------------------
*/

#include <stdio.h>
                 #include <stdlib.h>
#include <dir.h>
#include <string.h>
#include <dos.h>

/*
Only internal procedure used in this program. Since it requries
no input and returns nothing it is declared VOID for both in
going parameters and return values.
*/

void move (void);


/*
Global variables used throughout this program.
*/

char   searchstr[MAXPATH], newpath[MAXPATH],
       newname[MAXPATH],  oldname[MAXPATH],
          pathname[MAXPATH];
char   drive[MAXDRIVE], subdir[MAXDIR],
       file[MAXFILE], ext[MAXEXT];
int    done, count, problem;
struct ffblk dta;

/*
The main loop of this utility obtains arguments from the command
line and prints a copyright notice on the screen. If the command
line does not have enogh arguments, a help message is printed
and the program ends with an error code. If everything is alright,
then the string variable SEARCHSTR is set equal to the first parameter
and the second argument is copied to NEWPATH. Some  checking is done
on NEWPATH to see if it ends with a back slash (ASCII 92). If it does,
it is left alone. Otherwisse, one is added.

CALLED FROM:   (Dos)

CALLS:         move

*/

main (int argc, char *argv[])
{
  printf ("\nMV-Mov files, TC Edition\n");

  if (argc > 2)
    {
    strcpy (searchstr,argv[1]);
    fnsplit(searchstr, drive, subdir, file, ext);
    sprintf(pathname,"%s%s", drive, subdir);

    strcpy(newpath,argv[2]);
    count = strlen(newpath);
    if (newpath[count-1] != 92)
      {
      newpath[count] = 92;
      newpath[count+1] = 0;
      }
          }
  else
    {
    printf ("\nUsage: C:\>mv fromname.ext to path\n");
    return (1);
    }
  move ();
  return (0);     /* No problems encountered */
}

/*
This procedure searchs for matchs to the SEARCCHSTR and continues
to do so unit no more matches are found. All found files are moved
to the destination path specified buy NEWPATH with the file name
appended to it. This is accomlished below with the STRCAT command.

CALLED FROM:  ~main

CALLS:         (none)

*/

void move (void)
{
  count = 0;
  done = findfirst (searchstr, &dta, 47); /* all including hidden */

  while (!done)
    {
    strcpy (oldname, pathname);
    strcat (oldname, dta.ff_name);
    strupr (oldname);

    strcpy (newname, newpath);
    strcat (newname, dta.ff_name);
    strupr (newname);

    problem = rename (oldname, newname);

    if (!problem)
      {
      count++;
      printf ("\n  %-12s moved to %s",oldname, newname);
      }
    else
      printf ("\n\n %-12s not moved. ERROR!\n", oldname);

    done = findnext (&dta);
    }

 if (count > 0)
   printf ("\n\nNumber of files moved: %3d\n",count);
 else
   printf ("\nNO file(s) match.\n");

}
@
```